Memory Allocation Methods

Agenda

- Introduction
- First Fit
- Best Fit
- Worst Fit
- Conclusion

Introduction

Each time when malloc (memory allocation) or free is called you get pointer to the allocated memory address or free it back for others to use. Dynamic memory management requires frequent memory allocation and freeing up the space.

How to manage the freed and used space. How do we know which address/slots to be used while the next malloc call.
To answer above questions, there are three methods which we can used for managing the memory.

Lets discuss each method in detail.

Below is the current state of one large block of 1024bytes. We will see how each method uses for allocation of memory.

0 1023

Address	0	250	550	700	900	950 - 1023
Allocation Details	Block 1 allocated Size = 250	Free 1 Size = 300	Block 2 allocated Size = 150	Free 2 Size = 200	Block 3 allocated Size = 50	Free 3 Size = 73

Note: In this blog we would see only first bit algorithm. Each of these methods have their own use cases. But in general First Fit method is usually preferred.

First Fit

In the First Fit method, the free list is traversed sequentially to find the first free block which can hold the requested size.
Once we the block is found, one the below two operations are performed.

- If the size is equal to the requested amount, it is removed from the free list
- Else split into two parts. Here the first portion remains on the list and second is allocated. The reason of allocating the second part is the operations of updating the list could be avoided by just making change in size of the free node.

Lets now try to allocate 200, and see how the structure gets changed.

0 1023

Address	0	250	350	550	700	900	950 - 1023
Allocation Details	Block 1 allocated	Free 1	***Block 2 allocated***	Block 3 allocated	Free 2	Block 4 allocated	Free 3

	Size = 250	Size = 100	***(new)* *Size = 200***	Size = 150	Size = 200	Size = 50	Size = 73

As you could see the memory is allocated in the second part, leaving the previous block pointing to the first part. Else if we would have used first part, we need to make change in the list by pointing the previous to the second part.

Now lets see the algorithm:

```
p = freeblock;
alloc = null;  // pointer to store the allocated size n's address
q = null;

// find the free node which can allocate the given size n
while ( p != null && size(p) < n) {
   q = p;   // previous node
   p = next(p);
}

// if there is block large enough found
if ( p != null ) {
   s = size(p);
   alloc = p + s – n; // alloc contains the address of the desired block

   // if the block size matches the requested size, remove the block from the free list
   if ( s == n) {

      // if the match is found in the first block update the pointer of freeblock to point the next of
free block
      if ( q == null ) {
          freeblock = next(p);
      } else {
          next(q) = next(p);
      }
   } else {
     size(p) = s – n;  // adjust the size of the remaining free block
   }
}
```

Best Fit

In the Best Fit method, the smallest of the free block is choose whose size is greater than or equal to the requested size n. In this algorithm has to traverse the entire list to find the apt match.

Lets now try to allocate 200, and see how the structure gets changed.

0 1023

Address	0	250	550	700	900	950 - 1023

Allocation Details	Block 1 allocated Size = 250	Free 1 Size = 300	Block 2 allocated Size = 150	***Block 3 allocated (new) Size = 200***	Block 4 allocated Size = 50	Free 2 Size = 73

As you could see the memory Free 1 was ignored, Free 2 was updated to Block 3. Also now the Free 1 is pointing to Free 3 (now changed to Free 2). As the requested size matches the allocated size and its removed from the free pointer list.

Worst Fit

In the Worst Fit method, the algorithm always allocates the portion of the largest free block in memory. The logic behind this method that by using a small number of very large blocks repeatedly to satisfy the majority of the requests, many of the moderately sized blocks will be left unfragmented.

Lets now try to allocate 200, and see how the structure gets changed.

0 1023

Address	0	250	350	550	700	900	950 - 1023
Allocation Details	Block 1 allocated Size = 250	Free 1 Size = 100	***Block 2 allocated (new) Size = 200***	Block 3 allocated Size = 150	Free 2 Size = 200	Block 4 allocated Size = 50	Free 3 Size = 73

As Free 1 hold the maximum of 300, the memory is allocated there. Lets try to allocate 100, though we have first Free size with 100 it will still allocate in Free 2 which is the maximum now.

Conclusion

Each method has its own patterns. Lets see it with example.

First Fit is best case
In the below scenario only First Fit was able to serve all the requests.

Request	**Blocks remaining using**		
	First Fit	**Best Fit**	**Worst Fit**
Initially	110, 54	110, 54	110, 54
25	85, 54	110, 54	85, 54
70	15, 54	40, 29	15, 54
50	15, 4	Cannot be full filled	15, 4
14	1, 4		1, 4
1	0, 4		1, 3

Request	Blocks remaining using		
4	0, 0		Cannot be full filled

Best Fit is best case
In the below scenario only Best Fit was able to serve all the requests.

Request	Blocks remaining using		
	First Fit	**Best Fit**	**Worst Fit**
Initially	110, 54	110, 54	110, 54
50	60, 54	110, 4	60, 54
100	Cannot be full filled	10, 4	Cannot be full filled

Worst Fit is best case
In the below scenario only Worst Fit was able to serve all the requests.

Request	Blocks remaining using		
	First Fit	**Best Fit**	**Worst Fit**
Initially	200, 300, 100	200, 300, 100	200, 300, 100
150	50, 300, 100	50, 300, 100	200, 150, 100
100	50, 200, 100	50, 300, 0	100, 150, 100
125	50, 75, 100	50, 175, 0	100, 50, 100
100	50, 75, 0	50, 75, 0	0, 50, 100
100	Cannot be full filled	Cannot be full filled	0, 50, 0

As said, each have their own patterns. Though first fit method is generally preferred.

Reference
Data Structures using C and C++ by Yedidyah Langsam, Moshe J. Augenstein, Aaron M. Tenenbanum